Commodity computing results from the Swiss-Tx project

Ralf Gruber [a,*] Pieter Volgers [b] Alessandro De Vita [c]
Massimiliano Stengel [c] and the Swiss-Tx Team

[a] *SIC, Ecole Polytechnique Fédéral, CH-1015 Lausanne, Switzerland*
[b] *SMR Engineering Development, 2502 Bienne, Switzerland*
[c] *IRRMA, Ecole Polytechnique Fédéral, CH-1015 Lausanne, Switzerland*

Abstract

The aim of the Swiss-Tx project was to build, install, test and use high performance commodity computers. The biggest machine is the 70 Compaq Alpha processors Swiss-T1 machine installed at the EPFL computing centre and running in production mode since July 2000. This parallel MPI computer is well balanced in terms of processor speed, memory access, inter-processor communication network, and I/O capabilities. Due to the high flexibility of the cluster computing approach, it is today possible to fine tailor cost-effective MPI machines to the needs of real-life applications in engineering, database, WWW, data mining, or bioinformatics domains. Results obtained during the Swiss-Tx project are compiled in this paper.

Key words: cluster computing, supercomputing, commodity computing, MPI, bioinformatics, database, data mining

1 Introduction

Up to now application engineers had to adapt their parallel algorithms to the computer architectures. The computer manufacturers designed NUMA (Non Uniform Memory Access) parallel compute servers that include internal communication networks as powerful as possible. The operating system takes care of the data distribution considering the NUMA distributed memory as a virtually shared memory architecture. This choice eases use of a parallel computer,

* Corresponding author.
Email address: Ralf.Gruber@EPFL.ch (Ralf Gruber).

but its price is a few times higher than for a comparable distributed memory cluster machine made with commodity parts. If the maximum performance of a NUMA or of an SMP (Symmetric (Shared) Memory Processors) server is too small, clustering of servers is necessary, and the virtual shared memory advantage is lost.

Cluster commodity computers consist of PCs or workstations interconnected by networks that can be adapted to the needs of the applications. The programmer has to take care of the data distribution on the different processors as well as of the inter-processor data transfers by using the message passing library MPI (Message Passing Interface). Programs written in MPI show best efficiencies on clusters, and even on NUMA architectures they often run as fast or sometimes even faster than those using the easier programmable OpenMP library or threads. With the arrival of the cost-effective cluster machines, MPI programming could become the preferred parallel processing standard. An MPI program can easily be ported on other parallel platforms without loss of efficiency. In addition, cluster computing offers the new opportunity to tailor computer architecture to the communication needs of applications.

Another advantage of the cluster architecture is its flexibility. A cluster can grow to thousands of processors, not all processors have to be of the same type, or they even have not to come from the same computer manufacturer. Thus, one can start with a small initial configuration and regularly add new computational modules according to the user needs. These modules include the most recent technology, the newest, most powerful processors, memory units, disks, and communication networks. The job distribution is made by a resource management system running on the frontend on which a user prepares his job and gets the results back. The most often usage of parallel machines are data transaction applications such as Web service access, data mining, or in bioinformatics. In these domains, data distribution and collection have to be performed in the most efficient manner. Since the communication needs are small, these domains are predestinated for cluster computers.

A few years ago, EPFL has decided to prepare this new domain of cluster computing by launching the Swiss-Tx project that aimed at developing a parallel computer with commodity components interconnected with a high speed, low latency communication network [1–4]. These clusters have to fulfil all the expectations one imposes to parallel machines running in production mode in a computing centre. The cluster frontend must show high availability, there must be enough user and application disk space with efficient access capabilities [5,6], the computational unit has to be powerful and usable transparently. The major result of this project is the 70 Alpha processor Swiss-T1 parallel computer running in production mode since August 2000 at the EPFL [2]. Its high speed, low latency network, called TNet, with a specially optimised MPI communication library, comes from Supercomputing Systems [27]. It also

has a cost-effective, but less powerful, Fast Ethernet communication system and can operate like a Beowulf [7]. Different user communities have tested the machine [8–17] and validated the novel computer architecture.

Similar projects have been realised world-wide. Most of them are based on a pure Beowulf concept. Most closely related to the Swiss-Tx project is the scalable computing system C-Plant at Sandia National Laboratories [18] in which more than thousand Alpha processors are connected by a Myrinet switching network [19].

This paper starts with the definition of the very useful γ factors that characterise a parallel cluster machine and an algorithm. They enable tailoring of the computer architecture to the application. More detailed studied on this subject can be found in [20,21]. The Swiss-Tx project is then presented with a special emphasis on the 70 Compaq Alpha processors Swiss-T1 cluster running productively in EPFL's computing centre.

2 The γ factors

To study tailoring computer architectures to application needs, quantities have been introduced that relate processor and network performances. The γ_{mac} factor characterises a computer architecture and γ_{alg} characterises an algorithm. Their definitions are:

$$\gamma_{mac} = \frac{\text{effective processor performance [Mflop/s]}}{\text{effective bandwidth per processor [Mword/s]}} \tag{1}$$

$$\gamma_{alg} = \frac{\text{number of operations [Mflops]}}{\text{amount of data to transfer [Mwords]}} \tag{2}$$

2.1 The γ_{mac} factor

The γ_{mac} factor indicates how many operations can be executed during the transfer of one (64-bit word) operand over the network. The value of γ_{mac} depends on the effective performance of a processor which itself depends on the type and the local implementation of the algorithm. This effective processor performance depends on the memory access needs of the locally executed task. For real-life applications RISC processor efficiency is not very high. Typically, for the most efficient SGI O3K and Compaq Alpha processors average performances of around 20 to 25% of the peak performance can be measured. These relatively low efficiencies are directly related to the insufficient memory bandwidth of a processor. Main memory access should be reduced and access

of data in cache increased. For instance, matrix operations using the ScaLAPACK library [22] can reach up to 90% of the peak performance. These high efficiencies are due to the special cache optimisation effort made by all the computer manufacturers to reach highest performance in the Linpack benchmarks that position the computers in the Top500 list [23]. A programmer can profit from this performance race, and by an intensive use of these library modules he can develop highly efficient applications. In addition, these programs keep their efficiencies when they are ported on another computer platform.

The effective bandwidth per processor depends on the bandwidth per link, on the network topology [2] and on the network protocol. Today, the maximal bandwidth of a Fast Ethernet is 12.5 MB/s, 100 MB/s on a Gigabit Ethernet, and is over 200 MB/s on a Myrinet [19] and on a Quadrics network [24] . The network speed will strongly increase in 2002 by the arrival of new communication standards such as PCI-X and Infiniband, both increasing the peak bandwidth by a factor of four or more. For many years, network speed has not evolved rapidly enough, now, it will increase faster than the processor performance. These peak bandwidths cannot always be approached. For instance, it is known that the effective bandwidth of the Gigabit Ethernet is only a fraction of its peak performance and its latency is not smaller than the one of a Fast Ethernet. This is mainly due to the complex TCP/IP protocol. The Quadrics, Myrinet and TNet networks directly write from memory to memory, thus reducing the latency by at least a factor of ten with respect to Fast Ethernet, and reaching effective bandwidths close to peak.

Another important point is the network topology. A detailed study on their characteristics has been made in [2]. The two most promising network architectures are the Fat Tree as used in the Sierra/Compaq machines with Quadrics switches, and the Circulant Graph as used in the Swiss-T1.

Typical γ_{mac} factors are close to one for the NEC SX-5 vector machine, 8 for an SGI Origin 2000, 100 for a Beowulf with 64 dual processor Pentium III computers, interconnected with a Fast Ethernet switch. On the Swiss-T1, γ_{mac} is around 40 when using the TNet and 400 when passing the messages over the Fast Ethernet.

2.2 The γ_{alg} factor

In optimised parallel algorithms, the communication needs are minimised. It is highly recommended to also minimise the number of messages sent such that the latency in the communication network does not impair the communication speed. Both demands can be satisfied by choosing the standard communication library MPI, designed for message passing between processors

in a parallel, distributed memory computer architecture. The programmer himself has to take care of the data organisation and of the data transfer over the network. In a cluster computer, there is no single image operating system that supports data handling as in a NUMA machine. In such a parallel server, the system considers the memory, in reality physically distributed over all the processors, as to be one (virtually) shared global memory and it takes care of data distribution. This facilitates the use of parallel machines, but the overall efficiency of the application can suffer. In fact, the system attributes local memory in priority. However, depending on the memory demands of the co-users of a NUMA machine, the memory could more and more be attributed to far away processors. This increases the memory access time by a factor up to the memory speed over the internal interprocessor network speed. This factor is typically three and more, and, comparing with MPI implementations where memory locality is enforced, performance losses of up to a factor of two can be measured. In addition to this performance loss due to ease of programming, NUMA machines are restricted in the number of virtual shared memory processors. If this number has to be increased, a NUMA clustering has to be made that forces its users to switch to MPI programming. It is therefore highly recommended to design parallel applications for distributed memory cluster architectures and use the standard MPI communication library.

Shmem is another message passing library available on SGI and Cray computers. Shmem guarantees high bandwidth, memory locality, a very small latency and a direct memory to memory message transfer. Its use is simple, but unfortunately Shmem is not a standard.

The number of data transfers between processors strongly depends on the type of algorithms. When studying its dependence on the number of processors, different considerations have to be made. One can fix the size of the case and increase the number of processors. One can fix the number of processors and increase the problem size. One can fix the problem size per processor and increase the overall problem size and, accordingly, the number of processors. We opt for the last option, thus we will fix in our considerations the memory size and increase the size of the case and the number of processors. The γ_{alg} evolution as a function of the number of processors is studied for three application types: the so-called "embarassingly parallel" applications, a finite element approach, and a Fast Fourier Transform (FFT) algorithm for a Car-Parinello method in material sciences.

2.2.1 "Embarassingly parallel" algorithms

In "embarassingly parallel" algorithms, there is only communication between a master and a number of slave processors. No inter-processor communication is needed between the slaves. Examples are database and web transactions or

sequencing algorithms in genomics and proteomics. In such algorithms, a huge number of independent transaction-type of cases is sent from a master machine to the different slave processors and the results are collected. Communication needs are small and cost effective Fast Ethernet networks are sufficient.

2.2.2 Finite element approach

Finite element approaches based on domain decomposition are typical methods that are dominated by point-to-point communication between nearest neighbour subdomains. In a 3D finite element program, the number of operations is proportional to the total number of mesh points, to the square of the number of unknowns per mesh point, to the number of non-zero matrix elements, and to the number of operations per matrix element. The number of data to be transferred between processors is proportional to the number of mesh points on a surface, to the number of surfaces, and to the number of unknowns per mesh point. As a consequence, γ_{alg} for a finite element program is proportional to the number of mesh points in one direction, to the number of unknowns per mesh point, to the number of non-zero matrix elements, and to the number of operations per mesh cell, and inversely proportional to the number of surfaces. If one fixes the local problem size and increases the overall size and, accordingly, the number of processors, the γ_{alg} value remains strictly constant since all quantities entering γ_{alg} are constant. For real-life finite element approaches with millions of mesh points and with a few unknowns per mesh point, this γ_{alg} factor reaches quantities of several hundreds. This implies that such finite element approaches scale well on Beowulf machines and do not really request high performance network capabilities. If one fixes the problem size and increses the number of processors, the γ_{alg} value decreses with the volume/surface ratio, or with $\sqrt[3]{p}$. If the number of processors is fixed and the problem size increased, the γ_{alg} value is increased with the volume/surface ratio, or with $\sqrt[3]{p}$.

2.2.3 First principles chemistry applications

First principles chemistry applications tackle a class of fundamentally "entangled" problems, which can be highly communication intensive due to the underlying quantum formalism. The γ_{alg} factor is estimated for a quantum chemistry application, using a model momentum space algorithm in which each electron state is expanded on a basis of plane waves. They are easily evaluated by passing from real to reciprocal space by means of (Fast) Fourier Transformations (FFT). Energy minimisation algorithms are iterative, and basically involve dense linear algebra operations such as matrix algebra and diagonalisations (performed by BLAS and PBLAS subprograms) besides the mentioned FFTs. We assume the parallelisation strategy to be based on dis-

tributing the FFT mesh across the PEs. To evaluate the typical communication workload, data communication is assumed to be represented by 3D-FFTs per iteration, each involving two point-to-point global communications of the entire electronic orbital data set. Each orbital is represented on a cubic FFT mesh of a system dependent variable size. As test cases, we chose a communication intensive, Fourier dominated problem (128 electron states on a 112^3 grid) and a linear algebra dominated problem (432 electron states, on a 128^3 grid). Under the hypothesis above, the communications involve 4.3 and 21.8 Gbytes/iteration for the two systems, respectively. To estimate the CPU operation workload, we assume (i) a typical production run on 32 PEs delivering 200 Mflops/PE sustained on the Swiss-T1 machine and (ii) a realistic benchmark wall clock time per iteration of 3.8 and 48 seconds for the two systems, respectively. This leads to $24 \cdot 10^9$ and $300 \cdot 10^9$ floating point operations per iteration. The final γ_{alg} values for the two representative test systems considered are therefore 45 and 110 (floating point operations per eight-byte word communicated). FFT demands global communication operations such as all-to-many operations and, as a consequence, the γ_{alg} value is small. This means that first principles chemistry applications demand very powerful communication systems. Results for this tough problem are given at the end of this paper. A similar communication-demanding algorithm is the matrix transposition.

For these algorithms, the γ_{alg} value is almost independent of the problem size and the number of processors. In fact, if the local problem size is fixed and the number P of processors varies with the overall problem size, the γ_{alg} value varies as $\log(P)$. This implies that γ_{alg} increases slightly when the problem size increases. If the number of processors is fixed and the problem size increases, γ_{alg} slightly increases with $\log(N)$, where N corresponds to the number of Fourier terms. If one fixes the problem size and increases the number of processors, γ_{alg} remains constant.

2.3 Cluster tailoring condition

Let us define the ultimate γ factor and the cluster tailoring condition:

$$\gamma = \gamma_{alg}/\gamma_{mac} > 1 \tag{3}$$

This condition implies that the machine communication has to have a lower γ_{mac} value than γ_{alg} of the algorithm. In this case, the network is sufficiently powerful to not dominate the overall turn-around time. The influence of the latency time is not included in this condition.

For point-to-point communication dominated algorithms, such as in the finite element approach, γ_{alg} is high, the requested bandwidth of the communication

network can be small, and, as a consequence, Fast Ethernet switch connections are of sufficient performance. For global communication operations dominant algorithms such as the FFT, the communication bandwidth has to be high and the network topology chosen such that no bottlenecks appear. In this case, high performance networks like Myrinet, Quadrics, or TNet have to be chosen.

3 The Swiss-Tx project

3.1 The machines

The Swiss-Tx commodity parallel computer project was a co-operative work between the academic partners EPFL, ETHZ and CSCS and the industrial partners Compaq and Supercomputing Systems in Zurich. It aimed at integrating rack-mount Compaq DS20E Alpha-based shared memory dual processor computational units. These most powerful individual computers include the Compaq Tru64 UNIX operating system, Fortran, C and C++ compilers, and mathematical libraries. All the necessary hardware and software products to interconnect these computational units and integrate them to become a supercomputer have been developed during the project, or have been purchased on the market from independent software vendors.

During a first step, two prototype machines have been constructed to test the concept. Then, two now productive parallel clusters have been designed, built and installed at EPFL. One of them is a 16 processor departmental computer, the other one is the Swiss-T1.

The Swiss-T1 cluster (see Fig. 1) consists of three different major parts, the 64 processors computational unit, the 4 processors Frontend, and the 2 processors development unit:

- The computational unit has 32 dual processor Alpha-based DS20E computers with one Gigabyte (1 GB = 10^9Bytes) of main memory and 18 GB of disk space and a Compaq Tru64 UNIX operating system. Each processor runs at 500 MHz, delivering up to two results per cycle. These SMP computers are interconnected by the fast communication network TNet, developed by Supercomputing Systems, Zurich, and described later on, and by Fast Ethernet switches that also connects the computational unit to the Frontend. MPICH can be used to send messages over the Fast Ethernet. A specially developed and optimised MPI is used to send messages over the TNet.
- The Frontend unit consists of two DS20E computers interconnected with

a high speed Memorychannel link and running Trucluster UNIX operating system. Each DS20E sees the RAID disk unit of 300 GB in a fully symmetric manner. If one computer is down, the other takes over its work. This guarantees a 99.985% availability of the Frontend. Users enter through this Frontend, prepare their job and submit it to the computational unit by the GRD/Codine resource management system. The Totalview software can debug parallel jobs that use the MPICH communication library and run over the Fast Ethernet.

- The development unit is used to test new software before porting it onto the Frontend and the computational unit.

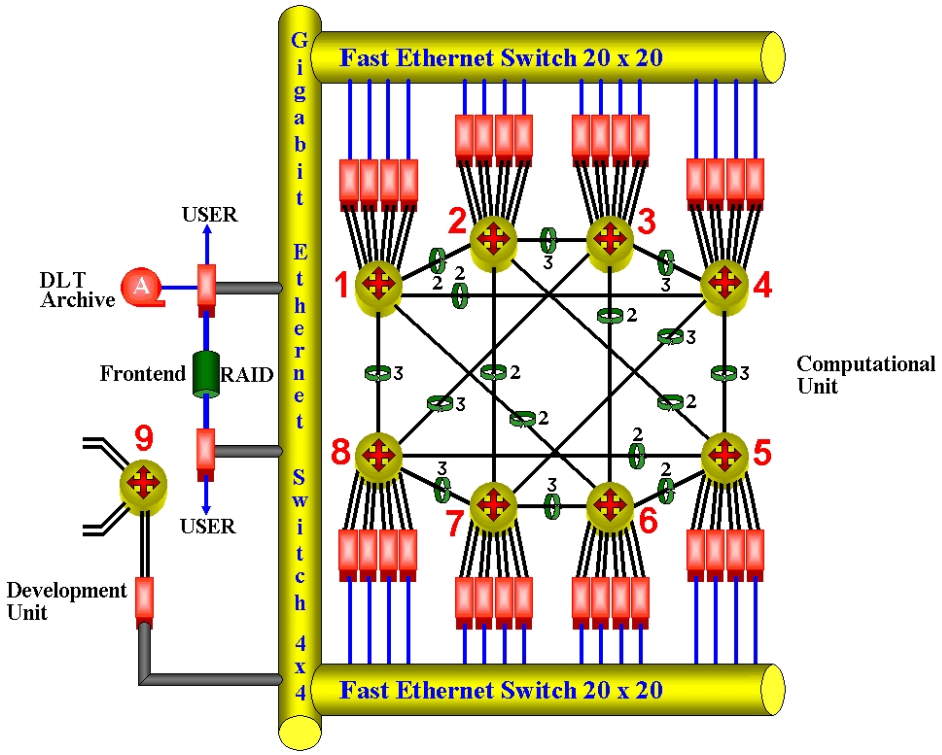

Fig. 1. The architecture of the Swiss-T1 with its eight 12x12 crossbars and the Fast Ethernet switch connection. The numbers on the links of the Circulant Graph network architecture denote the number of data transfers for an all-to-all operation between the crossbars.

3.2 The TNet network

The TNet network consists of the following components.

- 12x12 crossbar switches with bi-directional link speeds of up to 264 MB/s
- 32-bits / 33 MHz PCI adapters with a maximum bi-directional bandwidth of 132 MB/s

- FCI (Fast Communication Interface), a basic, very low, 5μs latency memory-to-memory data transfer message passing library
- MPI (Message Passing Interface) communication library, based on FCI with 15 μs latency
- Optimal routing table support for point-to-point and multicast operations
- Cosmos network management system

The network topology is based on the Circulant Graph concept with two K-rings [2]. The first ring connects the crossbars number 1 to 8, in the second ring every third crossbar is connected. The distance between two crossbars is one or two. In the case of distance two, four paths are possible between two crossbars. This gives the possibility to optimise the routing for multicast operations. For the Swiss-T1, the routing has been optimised for an all-to-all operation. When sending messages from any crossbar to any other one, the paths have been chosen in such a way that each link is used at most three times. The numbers on the links exactly indicate the link occupation for such an All-to-All operation. The Swiss-T1 machine has been installed at EPFL in January 2000 and runs in production mode since August 2000. A smaller test machine, the Baby-T1 with 16 Alpha processors and with one TNet crossbar switch runs productively as a departmental machine.

3.3 The Software

In the Swiss-T1 machine, all software in the computational commodity unit DS20E originates from Compaq, all the software needed to connect these units come from Independent Software Vendors (ISV) or have specially been developed during the project. The resource management software GRD/Codine, now made public, distributes the jobs among the DS20E processors. Totalview enables debugging of the jobs when running the message passing interface MPICH over the Fast Ethernet. A special Striped File I/O system SFIO [5,6] has been developed and is available on the Swiss-T1. A great number of user applications [8–15] were ported to test the parallel cluster and to run productively on it. The engineering visualisation system Ensight has been ported and tested [16]. Ensight is able to read data coming from commercial Computational Fluid Dynamic (CFD) programs such as Fluent [12] and from the data management system Memcom [17] used in the applications [13,14].

4 Results with the LAUTREC program

Using the Car-Parrinello package LAUTREC [28], on a medium water system, tests have been performed on different NUMA, MPP and cluster machines.

They show that up to 16 processors, the Swiss-T1 cluster (1'000 Mflop/s peak per PE) is about as powerful as the IBM SP3 (1'500 Mflop/s peak per PE) and the SGI Origin 3K (800 Mflop/s peak per PE) NUMA machines, and about 70% more powerful than a Cray T3E (1'200 Mflop/s per PE).

We have to mention here that the γ_{alg} value of the case is 45, independent of the number of processors. Thus, the Swiss-T1 with its γ_{mac} value of 40 has just a sufficient communication performance. A Fast Ethernet network with a much higher γ_{alg} value gives very poor performance.

In the case of a non-linear finite element computation reported in [29], ideal speedups have been observed on up to 64 processors.

5 Conclusions

The emerging cluster computer technology opens the new opportunity to tailor the computer architecture to the needs of the user applications. This technology is highly flexible, reduces the dependence of the customer on a computer manufacturer. The machines can be upgraded with the newest and most powerful commodity components of the market. The use of MPI gives to a customer the opportunity to easily move from one type of processors to another one without loss of message passing performance. The customer gets more freedom in choosing the best suited parallel computer. The cluster architecture can grow in an inhomogeneous manner. Thus, a customer can immediately profit from the arrival of new more powerful commodity parts that can be used to form a new and more powerful cluster unit leaving the old one untouched. Resource management systems exist to build inhomogeneous clusters, even with processors coming from different sources. This enables tailoring the computer architecture to each individual step of a full application. Due to the use of commodity components, cluster supercomputers are highly cost-effective. In addition, old cluster parts can be redistributed as individual workstations and reused for additional few years.

The Swiss-T1 machine with its Circulant Graph network realised with the high quality and efficient TNet and operated through a very efficiently implemented MPI show network characteristics that seem to be sufficiently powerful to run applications demanding high bandwidth and low latency. The frontend system has high availability. It enables preparing, testing and submission of jobs and result diagnostics.

The know-how accumulated in the Swiss-Tx project is being commercialised within the Start-up Company ClusterSolutions SA recently created in Lausanne [25].

Acknowledgements

The Swiss-Tx project has been financed by the CTI (Commission for Innovation and Technology). A special acknowledgement goes to all the over 80 persons who contributed with small to total of their time during the three years period. Thanks go also to all our American friends who meet with us once per year at the Commodity Computing workshops [26] and who helped us a lot to make the right choices.

References

[1] R. Gruber and A. Gunzinger, "The Swiss-Tx Supercomputer Project", Speedup 11 (1997) Number 2, p.20–26, http://capawww.epfl.ch/swiss-tx and http://tone.epfl.ch

[2] P. Kuonen and R. Gruber, "Parallel Computer Architectures for Commodity Computing", EPFL Supercomputing Review 11 (1999) 3–11

[3] S. Brauss, M. Frey, A. Gunzinger, M. Lienhard and J. Nemecek, "Swiss-Tx Communication Libraries", Proc. of HPCN99 (Amsterdam, 12-14 April, 1999), 10

[4] Stephan Brauss, "Communication libraries for the Swiss-Tx machines", EPFL Supercomputing Review 11 (1999) 12–15

[5] B. Gennart, E. Gabrielyan and R.D. Hersch, "Parallel file striping on the Swiss-Tx architecture", EPFL Supercomputing Review 11 (1999) 15–22

[6] Emin Gabrielyan, "SFIO, Parallel file striping for MPI-I/O", EPFL Supercomputing Review 12 (2000) 17–21

[7] http://www.beowulf.org

[8] A. S. Röhrl, "Fast, Portable, Predictable and Scalable Bootstrapping", Proc. of Interface98

[9] J.-A. Ferrez, K. Fukuda, Th. M. Liebling "Parallel Computation of the Diameter of a Graph on the Cray T3D, the SGI Origin 2000, the Swiss-T0, and with MPI", High Performance Computing and Applications (HPCA'98), Edmonton, Canada, May 20-22, 1998 Int. Conf. on Operations Research and annual meeting of GOR, ÖGOR and SIGOPT, ETH Zurich, 31 August - 3 September 1998

[10] J.-A. Ferrez, K. Fukuda and T.M. Liebling, "Parallel Implementation of Graph Diameter Algorithms", EPFL Supercomputing Review 10 (1998) 3–6

[11] Daniel Weill, "Test de performance du code Speculoos sur l'ordinateur parallèle T0-Dual", EPFL Supercomputing Review 11 (1999) 29–32

[12] M. L. Sawley, T.-M. Tran and T.L. Tysinger, "The Fluent 5 benchmark results on the Swiss-T1", EPFL Supercomputing Review 12 (2000) 13–16

[13] Pieter Volgers, "HPC in computational structural mechanics", EPFL Supercomputing Review 12 (2000) 29–31

[14] Jan B. Vos, "Flow simulations on high performance computers using the NSMB flow solver", EPFL Supercomputing Review 12 (2000) 32–40

[15] Y. Dubois-Pèlerin, R. Gruber and the Swiss-Tx Team, "The Swiss-Tx supercomputer project", ERCOFTAC journal, 1998

[16] Jean M. Favre, "Visualisation tools and environments for very large data", EPFL Supercomputing Review 12 (2000) 9–12

[17] http://www.smr.ch

[18] http://www.cs.sandia.gov

[19] http://www.myri.com

[20] R. W. Hockney and C. R. Jesshope, "Parallel Computers 2" (Adam Hilger, 1988)

[21] Jan Foster, "Designing and Building Parallel Programs" (Addison Wesley, 1994)

[22] http://www.netlib.org/scalapack/

[23] http://www.top500.org

[24] http://www.quadrics.com

[25] http://www.clustersolutions.com

[26] http://www.cs.sandia.gov/Conferences/SOS

[27] http://www.scs.ch

[28] A. De Vita, A. Canning, G. Galli, F. Gygi, F. Mauri and R.Car, "Quantum molecular dynamics on massively parallel computers", EPFL Supercomputing Review 6 (1994)

[29] Pieter T. G. Volgers, " High performance explicit transient structural analysis", Thesis 2200, EPFL, 2000